A STORAGE-EFFICIENT METHOD FOR CONSTRUCTION OF A THIESSEN TRIANGULATION

A. K. Cline

Department of Computer Sciences

University of Texas at Austin

R. J. Renka

Computer Sciences Division

Union Carbide Corporation - Nuclear Division

Oak Ridge National Laboratory

ACKNOWLEDGEMENTS

This report was extracted, with minor revisions, from the second author's PhD dissertation presented to The University of Texas at Austin in May, 1981. The first author's research was partially supported by NASA Grant NAG1-79.

ABSTRACT

This paper describes a storage-efficient method and associated algorithms for constructing and representing a triangulation of arbitrarily distributed points in the plane.

1. INTRODUCTION

This paper addresses the problem: Given a set of nodes (X_i,Y_i), $i = 1,\ldots,N$, arbitrarily distributed in the X-Y plane, construct a triangulation with the nodes as vertices and which is as nearly equiangular as possible. The primary application of such a triangulation is as a preliminary step in a triangle-based method for bivariate interpolation of data values associated with the nodes.[1] The triangulation also serves as an efficient mechanism for solving closest-point problems such as finding the two closest nodes and finding a largest circle containing none of the nodes. These problems arise in a variety of applications; e.g., wire layout, clustering, facilities location, and constructing the feasible polygon for linear programming in two variables with N constraints.[2] Another application of the triangulation method is as an automatic mesh generator for a triangle-based finite element code. Lists of element node numbers and boundary node numbers can be generated from a set of nodal coordinates which are concentrated in regions where the solution varies most rapidly. In addition to generating the mesh, the method guarantees its validity. Simpson provides and discusses the importance of an algorithm which verifies the consistency of a set of finite element input data.[3]

This paper describes the algorithms implemented in an extensive and well-documented software package for triangulation and interpolation written in a subset of ANSI standard FORTRAN accepted by the PFORT verifier.[4] The software listing is found in Renka[8] and machine-readable code may be obtained from the second author. The primary goal of this research was a triangulation package requiring less than those previously available. This was achieved at the cost of a relatively small loss in time efficiency over alternative packages.

Section 2 presents definitions and theory necessary to describe the algorithms. A more detailed analysis is found in Renka.[12] Section 3 describes the algorithms for constructing a Thiessen triangulation. This presentation is independent of a data structure for representing the triangulation. In Section 4 several data structures are analyzed including one which requires only 7 words of storage for each node. Execution timings were also performed and are discussed in Section 5.

2. FUNDAMENTAL DEFINITIONS AND TRIANGULATION THEORY

Definition 1. For three points $P_0=(X_0,Y_0)$, $P_1=(X_1,Y_1)$, and $P_2=(X_2,Y_2)$ in the plane with $P_1 \neq P_2$,

P_0 LEFT $P_1 \rightarrow P_2$ if and only if $(X_2-X_1)\,(Y_0-Y_1) \geq (X_0-X_1)\,(Y_2-Y_1)$.

Thus we say P_0 is left of $P_1 \rightarrow P_2$ if and only if P_0 is to the left of or on the line through P_1 and P_2 as seen by an observer at P_1 facing P_2. Also we say P_0 STRICTLY LEFT $P_1 \rightarrow P_2$ if the inequality is strict.

Definition 2. A region of the plane is convex if and only if for any two points contained in the region the line segment connecting the two points also lies in the region. The convex hull of a finite set of points in the plane is the smallest convex region which contains the points. Note that such a convex hull is closed and thus contains the points which define it, and the convex hull of a finite set of noncollinear points has at least three of the points on its boundary.

Let S be an ordered set of distinct points in the plane (X_i,Y_i), $i = 1,2,\ldots,N$ where $N \geq 3$. It is assumed that the points are not all collinear. Let H be the convex hull of S and let B be the boundary of H.

Definition 3. A node is an element of S. Each node is identified with its index in S (node $i = (X_i,Y_i)$) and will be denoted by this index or by N_0, N_1, N_2,... indicating arbitrary indices. The distinction between a node and its index will be clear from the context.

Definition 4. A triangle is the convex hull of three noncollinear nodes, referred to as vertices. The triangle with vertices N_1, N_2, and N_3 is denoted (N_i,N_j,N_j) with N_k STRICTLY LEFT $N_i \rightarrow N_j$ for $i,j,k \in \{1,2,3\}$ and pairwise disjoint.

Definition 5. A triangulation of S is a set of triangles T with the following properties:

a) Each triangle contains no node other than its vertices,

b) The interiors of the triangles are pairwise disjoint, and

c) T covers H; i.e., every point in H is contained in a triangle of T.

Definition 6. Let T be a triangulation of S: for each triangle (N_1,N_2,N_3) of T, the vectors $N_1 \rightarrow N_2$, $N_2 \rightarrow N_3$, and $N_3 \rightarrow N_1$ are referred to as edges of T. The undirected line segment corresponding to an edge $N_1 \rightarrow N_2$ is referred to as an arc and is denoted $N_1 - N_2$ or $N_2 - N_1$. Note that edges and arcs are associated with a triangulation, and edges are directed while arcs are not.

A boundary edge of T is an edge $N_1 \rightarrow N_2$ such that P LEFT $N_1 \rightarrow N_2$ for all P$\notin$H. Thus each point of a boundary edge lies on B. An interior edge is an edge which is not a boundary edge. The arc associated with a boundary (interior) edge is a boundary (interior) arc.

A boundary node is a node which is an endpoint of a boundary edge and thus lies on B. An interior node is a node which is not an endpoint of a boundary edge.

The node opposite an edge $N_1 \rightarrow N_2$ is the node N_3 such that (N_1,N_2,N_3) is a triangle of T. N_3 is unique by Definition 5b.

Definition 7. Two nodes are adjacent to each other if and only if they are the endpoints of a common arc. A neighbor of a node N_0 is a node which is adjacent to N_0. The degree of a node N_0 is the number of neighbors of N_0.

The adjacency set for a node N_0 is an ordered set containing the neighbors of N_0 in counterclockwise order. The first neighbor is arbitrary if N_0 is interior. If N_0 is a boundary node, the first and last neighbors of N_0 are the unique boundary nodes N_F and N_L such that $N_0 \rightarrow N_F$ and $N_L \rightarrow N_0$ are boundary edges.

The following observations were made by Lawson.[6]

Theorem 1. Let T be any triangulation of S and

N = number of nodes,
N_b = number of boundary nodes = number of boundary arcs,

N_a = number of arcs,

N_t = number of triangles,

then

$$N_t = 2N - N_b - 2$$

$$N_a = N_t + N - 1 = 3N - N_b - 3$$

$$N - 2 \leq N_t \leq 2N - 5 \quad \text{and} \quad 2N - 3 \leq N_a \leq 3N - 6.$$

Note that the formula $N_a = N_t + N - 1$ is a special case of Euler's formula which holds for N_t representing the number of finite regions in any connected planar graph.

Definition 8. Given a triangulation T of S and a point P in the plane, a node N_0 is visible from P if and only if P can be connected to N_0 by a line segment which neither intersects nor overlaps an arc of T. (Line segments which meet in a "T" are said to intersect while segments which meet in a "V" do not.) P is said to be exterior to T if and only if P is not contained in the convex hull of S.

Definition 9. A pair of triangles of T, (N_1,N_2,N_3) and (N_2,N_1,N_4), which share a common arc form a quadrilateral of T denoted (N_1,N_2,N_3,N_4). The quadrilateral is said to be strictly convex if the diagonals $N_1 - N_2$ and $N_3 - N_4$ intersect at an interior point of $N_1 - N_2$. A swap in this case

is the replacement in T of (N_1,N_2,N_3) and (N_2,N_1,N_4) by (N_3,N_4,N_2) and (N_4,N_3,N_1). Thus, a swap in effect swaps diagonals.

A swap leaves the parameters N, N_b, N_a, and N_t unchanged, and all possible triangulations of S can be generated by applying swaps to a given triangulation. These results are proved in Renka.[12]

Definition 10. Given an interior arc $N_1 - N_2$ with corresponding quadrilateral (N_1,N_2,N_3,N_4), the swap test is a decision on whether to perform a swap based on either of the following criteria (which can be shown to be equivalent).

a) The max-min angle criterion is the choice of the pair of triangles which maximizes the minimum of the six interior angles when (N_1,N_2,N_3,N_4) is strictly convex. The decision will be positive if and only if a larger minimum angle would result from the swap in this case. The decision is defined to be negative if (N_1,N_2,N_3,N_4) is not strictly convex.

b) The circle criterion is the choice of the pair of triangles whose circumcircles do not contain the remaining vertices in their interiors. Equivalently, the decision will be made to swap if and only if N_4 is interior to the circumcircle of N_1, N_2, and N_3.

An arc is locally optimal if and only if it is a boundary arc or, if it is interior, application of the swap test to it would not result in a decision to swap.

Definition 11. A Thiessen triangulation is one in which all arcs are locally optimal.

Consider the following method for constructing a triangulation of S. For each node N_0 define the Thiessen region associated with N_0 to be the closure of the set of points in the plane which are closer to N_0 than to any other node. A pair of nodes N_1 and N_2 are said to be Thiessen neighbors

if and only if their corresponding Thiessen regions share one or more points. If the regions share more than one point, N_1 and N_2 are strong Thiessen neighbors. If the regions share exactly one point, N_1 and N_2 are referred to as weak Thiessen neighbors. This is the neutral case corresponding to four or more nodes lying on a common circle. A triangulation may be obtained by connecting all pairs of strong Thiessen neighbors and arbitrarily choosing $k - 3$ nonintersecting arcs connecting weak Thiessen neighbors when k nodes lie on a common circle for $k \geq 4$. Both Lawson[5] and Sibson[7] have proved that this triangulation satisfies Definition 11 above.

Figure 1 depicts a Thiessen region diagram with corresponding Thiessen triangulation T. Note that Thiessen region vertices are circumcenters of the triangles of T, and Thiessen region boundaries are composed of perpendicular bisectors of the arcs of T. Either $N_1 - N_2$ or $N_3 - N_4$ may be chosen as an arc in the neutral case depicted.

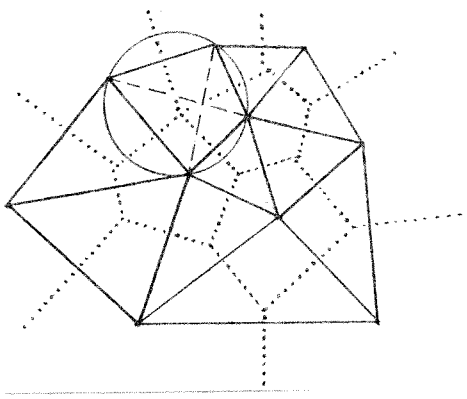


Fig. 1. Thiessen Region Diagram and Corresponding (Dual) Triangulation

Thiessen regions are also referred to in the literatures as tiles and polygons, and have been associated with the names Dirichlet and Voronoi. The corresponding triangulation is also referred to as a Delaunay triangulation. Rhynsburger[8] presents a brief history of the development of these concepts.

The triangulation algorithm which will be described in the following section applies the swap test in an iterative fashion to produce a Thiessen triangulation.

3. ALGORITHMS

This section discusses algorithms for constructing and obtaining information from a Thiessen triangulation. Some of the algorithms are described in more detail than others, but none of the descriptions make reference to a data structure. However, Algorithms 3, 5, and 7 are designed specifically for a data structure in which the adjacency information is explicit; i.e., the adjacency sets are easily obtained.

An overall description of the method will be followed by detailed algorithms. Our basic approach to constructing the triangulation is as follows:

a) Optionally, reorder the nodes (Algorithm 1).

b) Construct an initial Thiessen triangulation T_j of the first j nodes where j is the smallest integer such that nodes 1,...,j are not all collinear (Algorithm 2).

c) For k = j + 1,...,N construct a Thiessen triangulation T_k of nodes 1,...,k as follows:

   i) Find a triangle or a boundary arc of T_{k-1} which contains node k, or a pair of boundary nodes which are visible from k if k is exterior to T_{k-1} (Algorithm 3).

   ii) If k lies on a boundary arc, connect it to the endpoints. If k is contained in a triangle and does not lie on a boundary arc, connect it to the three vertices. Otherwise connect k to all boundary nodes which are visible from k (Algorithm 4).

   iii) Optimize the mesh by applying a sequence of swap tests (Algorithm 6) and the appropriate swaps to the interior arcs which are opposite k (Algorithm 5).

One of the features of this method is that it provides for efficient updating of the data. To add a new node to the triangulation, it is only necessary to execute Step c) an additional time, rather than recreating the entire triangulation.

Algorithm 1. Reorder the N nodes by applying an O(N*log(N)) quick sort to either their X or Y components.

Algorithm 1 may be employed at the user's option to increase the efficiency of the triangulation algorithms. It is not necessary but may be computationally advantageous even for small values of N. Timing comparisons for randomly ordered nodes versus presorted nodes are presented in Section 5.

The following algorithm initializes the triangulation. Except in the case of collinearity, it results in a single triangle, formed by the first three nodes as vertices.

Algorithm 2. Construct a Thiessen triangulation of nodes 1,...,j where j is the smallest integer such that the first j nodes are not all collinear. Such an integer exists by assumption and $3 \leq j \leq N$. Note that nodes 1,...,j-1 define a line segment L with two of the nodes as endpoints.

Step 1: Order the fist j - 1 nodes by their distances from one of the endpoints of L.

Step 2: Connect the first j - 1 nodes by making nodes N_1 and N_2 adjacent if and only if N_1 immediately precedes or immediately follows N_2 in the ordering defined in Step 1.

Step 3: Connect j to each of the first j - 1 nodes.

Clearly, the triangulation constructed by Algorithm 2 contains no strictly convex quadrilaterals and is thus a Thiessen triangulation. See Fig. 2.

The following algorithm may be used both to construct the triangulation and to locate a point at which an interpolated value is to be computed.

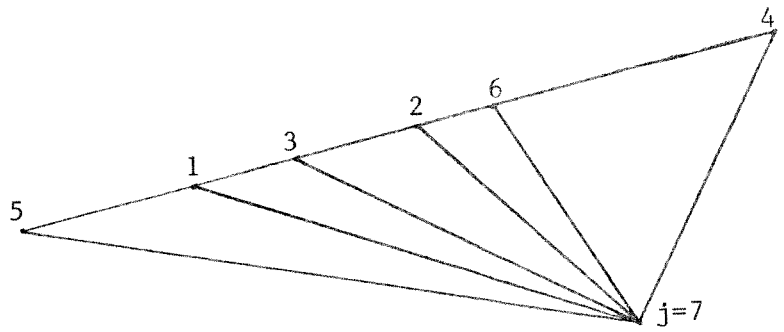


Fig. 2. Thiessen Triangulation Constructed by Algorithm 2.

Algorithm 3. Let T be a triangulation of S, let H be its convex hull, let B be its boundary and let P be any point in the plane.

a) If P is in H and does not lie on B, find a triangle (N_1,N_2,N_3) containing P (set N_1, N_2, and N_3 to the vertex indices of the triangle).

b) If P is exterior to T, set N_3 to 0 and find the indices of a pair of boundary nodes N_1 and N_2 such that N_1, N_2, and all boundary nodes encountered as the boundary is traversed from N_1 to N_2 in counterclockwise order are visible from P, no other nodes being visible from P.

c) If P lies on B, find the indices of the endpoints of a boundary edge $N_1 \rightarrow N_2$ containing P, and set N_3 to 0.

Let N_0 be an arbitrary node (index).

Loop 1. (Initialization.)

Step 1: Set N_F and N_L to the (indices of the) first and last neighbors of N_0, respectively.

Step 2: If N_0 is a boundary node and P NOT LEFT $N_0 \rightarrow N_F$, then set N_1 and N_2 to N_0 and N_F, respectively, and go to Step 13. If N_0 is a boundary node and P NOT LEFT $N_L \rightarrow N_0$, then set N_1 and N_2 to N_L and N_0, respectively, and go to Step 15.

Step 3: If there exists an arc $N_0 - N_2$ such that P NOT LEFT $N_0 \rightarrow N_2$, then determine N_2 and go to Step 5.

Step 4: If P coincides with N_0, then set N_0 to N_L; otherwise set N_0 to

the node opposite $N_L \rightarrow N_0$ (the neighbor of N_0 which immediately precedes N_L in the adjacency set). Go to Step 1.

Loop 2. (Find a cone with vertex N_0 and containing P.)

Step 5: Set N_1 to the node opposite $N_2 \rightarrow N_0$.

Step 6: If P Left $N_0 \rightarrow N_1$, then set N_3 to N_0 and go to step 5.

Loop 3. (Edge-hopping loop.)

Step 7: If P LEFT $N_1 \rightarrow N_2$, then go to Step 11.

Step 8: If $N_1 \rightarrow N_2$ is a boundary edge, then go to Step 13.

Step 9: Set N_4 to the node opposite $N_2 \rightarrow N_1$.

Step 10: If P LEFT $N_0 \rightarrow N_4$, then set N_3 to N_1 and N_1 to N_4; otherwise set N_3 to N_2 and N_2 to N_4. See Fig. 3. Go to Step 7.

Step 11: If $N_3 \rightarrow N_1$ is a boundary edge and P LEFT $N_1 \rightarrow N_3$, then set N_2 to N_1, N_1 to N_3, and N_3 to 0 and stop.

Step 12: If $N_1 \rightarrow N_2$ is a boundary edge and P LEFT $N_2 \rightarrow N_1$, then set N_3 to 0. Stop.

Loop 4. (Counterclockwise boundary traversal).

Step 13. Set N_B to the first neighbor of N_2.

Step 14: If P NOT LEFT $N_2 \rightarrow N_B$, then set N_2 to N_B and go to Step 13.

Loop 5. (Clockwise boundary traversal.)

Step 15: Set N_B to the last neighbor of N_1.

Step 16: If P NOT LEFT $N_B \rightarrow N_1$, then set N_1 to N_B and go to Step 15.

Step 17: Set N_3 to 0. Stop.

Several theoretical properties regarding this algorithm are proven in Renka[12] including that it correctly terminates after a finite number of operations. It is also shown that the number of iterations of Loop 1 is at

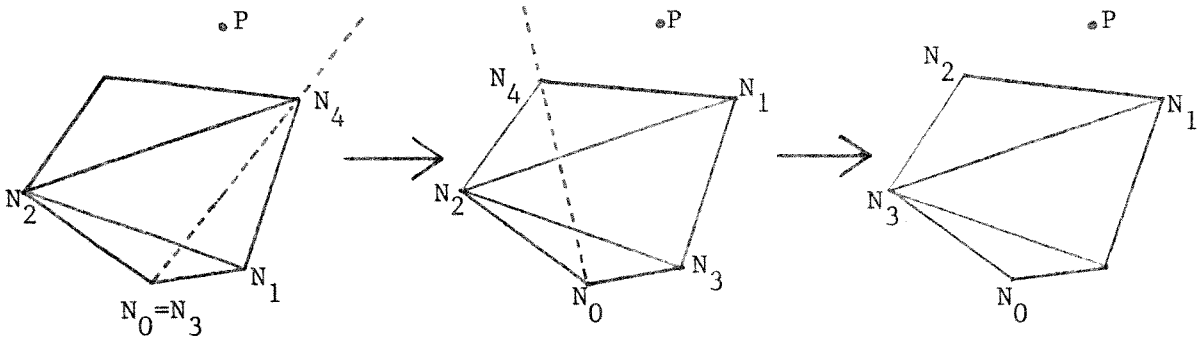


Fig. 3. Example of Loop 3.

most two, of Loop 2 is at most the degree of N_0, of Loop 3 is at most N_a (the number of arcs of T), and of loops 4 and 5 is at most N_b (the number of boundary nodes). Since $N_a = 3N - N_b - 3$, the operation count for the entire algorithm is bounded by an expression of the form $K_1 + K_2\ N + K_3\ N_b$. The actual operation count depends on the proximity of P to the starting node N_0. If a sequence of interpolation points are to be located, N_0 may be taken to be one of the nodes determined by the previous call. This gives good results when the points are ordered in some typical fashion, such as the natural ordering on a rectangular grid.

In constructing the triangulation, N_0 is chosen to be the node which was most recently added. If the nodes are ordered by Algorithm 1, the new node to be added, P, is always exterior to T and N_0 is always a boundary node which is visible from P. Thus the algorithm consists essentially of Loops 4 and 5 and has an operation count of the form $O(N_b)$. Our test results, however, indicate that the operation count does not increase with N_b.

Algorithm 4. Given a triangulation T_{k-1}, a new node to be added k, and nodes N_1, N_2, and N_3 determined by Algorithm 3:

a) If $N_3 \neq 0$, make k adjacent to N_1, N_2, and N_3.

b) If $N_3 = 0$, make k adjacent to N_1, N_2, and all boundary nodes en-

countered as the boundary is traversed from N_1 to N_2 in counterclockwise order.

The details of the algorithm depends on the data structure and are omitted. See Fig. 4.

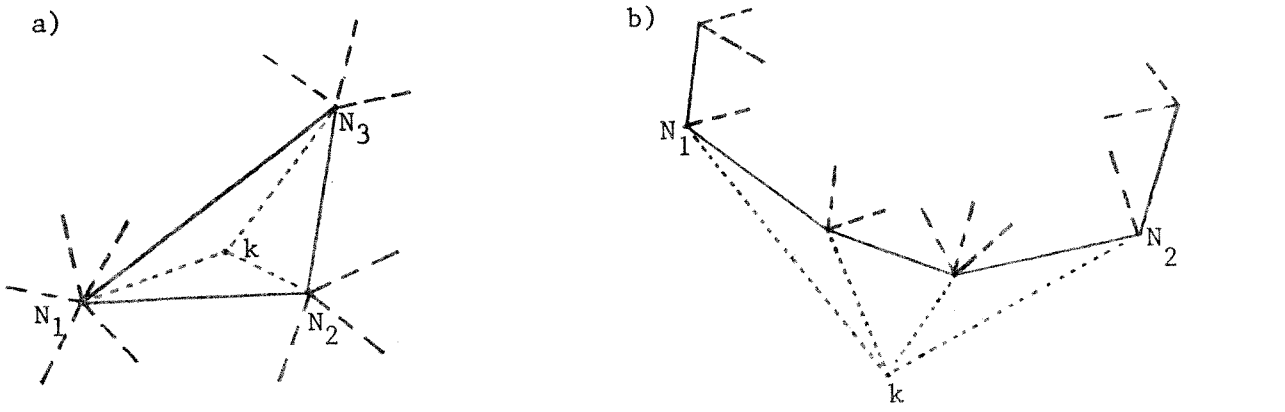


Fig. 4. Examples of Algorithms 4a and 4b.

Algorithm 4 clearly produces a triangulation of nodes 1,...,k when k does not lie on an arc. The case of k lying on an arc is discussed in Renka.[12]

Algorithm 5. Given a triangulation T_k constructed by adding node k (Algorithm 4) to a Thiessen triangulation T_{k-1}, optimize the triangulation:

Step 1: Set N_1 and N_F to the first neighbor of k.

Step 2: Set N_2 to the node opposite k $\rightarrow N_1$.

Step 3: If $N_1 \rightarrow N_2$ is a boundary edge, then go to Step 6.

Step 4: Set N_3 to the node opposite $N_2 \rightarrow N_1$.

Step 5: Test $N_1 - N_2$ for a swap. If the test is positive, then swap $N_1 - N_2$ for $k - N_3$, set N_2 to N_3, and go to Step 3.

Step 6: Set N_1 to N_2.

Step 7: If $N_1 = N_F$ or $N_1 \rightarrow k$ is a boundary edge, then stop; otherwise go to Step 2.

See Fig. 5.

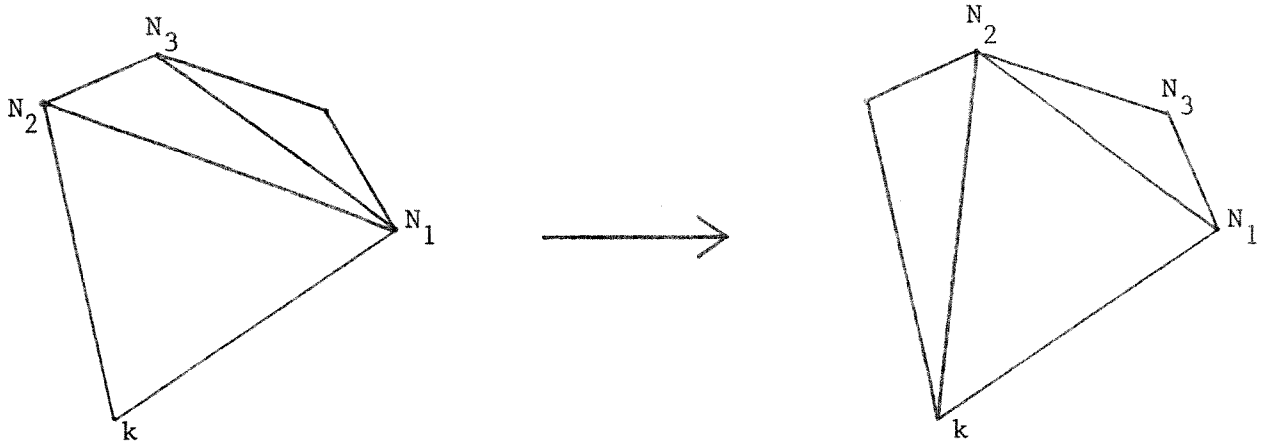


Fig. 5. Swap Applied by Algorithm 5.

The proof that Algorithm 5 produces a Thiessen triangulation is provided by Lawson.[5]

Algorithm 6a. Given the coordinates of the vertices of a quadrilateral of T, (N_1,N_2,N_3,N_4), set the logical variable SWPTST to TRUE if

$$SIN12 \equiv ((X_1-X_3)(X_2-X_3)+(Y_1-Y_3)(Y_2-Y_3))((X_2-X_4)(Y_1-Y_4)-(X_1-X_4)(Y_2-Y_4))$$
$$+ ((X_1-X_3)(Y_2-Y_3)-(X_2-X_3)(Y_1-Y_3))((X_2-X_4)(X_1-X_4)+(Y_2-Y_4)(Y_1-Y_4))<0.$$

Otherwise set SWPTST to FALSE. A swap is to be applied if and only if SWPTST is TRUE.

Theorem 2. Algorithm 6a produces the correct decision on the swap test (assuming computation is exact).

Proof: Let $\alpha_1 = \angle\ N_2\ N_3\ N_1$ and $\alpha_2 = \angle\ N_1\ N_4\ N_2$ (ref. Fig. 6). From geometry, the measure of the circular arc subtended by α_1 is smaller than twice the measure of α_2 if and only if N_4 is interior to the circle circumscribing N_1, N_2, and N_3. Furthermore the arc subtended by α_1 has twice

the measure of α_1. Since the sum of the measures of the arcs is 2π, we may conclude that N_4 is interior to the circumcircle (i.e., a swap should be performed) if and only if $\alpha_1 + \alpha_2 > \pi$. Equivalent to $\alpha_1 + \alpha_2 > \pi$ is $\sin(\alpha_1 + \alpha_2) < 0$ since $\alpha_1 + \alpha_2 < 2\pi$. Finally

$$\sin(\alpha_1 + \alpha_2) = \cos(\alpha_1)\sin(\alpha_2) + \sin(\alpha_1)\cos(\alpha_2)$$

$$= \frac{((X_1-X_3)(X_2-X_3)+(Y_1-Y_3)(Y_2-Y_3))}{((X_1-X_3)^2+(Y_1-Y_3)^2)^1((X_2-X_3)^2)+(Y_2-Y_3)^2)^1} \frac{((X_2-X_4)(Y_1-Y_4)-(X_1-X_4)(Y_2-Y_4))}{((X_2-X_4)^2+(Y_2-Y_4)^2)^1((X_1-X_4)^2+(Y_1-Y_4)^2)^1}$$

$$+ \frac{((X_1-X_3)(Y_2-Y_3)-(X_2-X_3)(Y_1-Y_3))}{((X_1-X_3)^2+(Y_1-Y_3)^2)^1((X_2-X_3)^2+(Y_2-Y_3)^2)^1} \frac{((X_2-X_4)(X_1-X_4)+(Y_2-Y_4)(Y_1-Y_4))}{((X_2-X_4)^2+(Y_2-Y_4)^2)^1((X_1-X_4)^2+(Y_1-Y_4)^2)^1},$$

and the identical denominator of these terms is positive. □

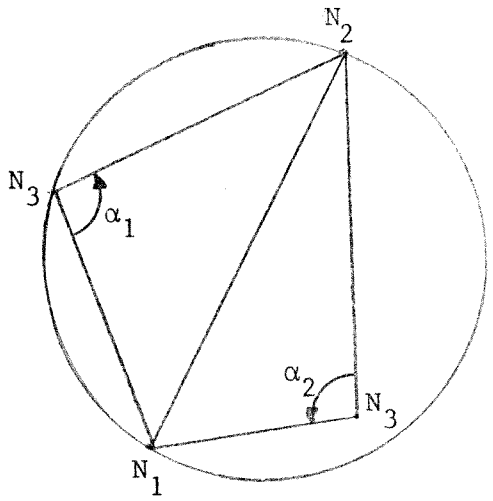


Fig. 6. Quadrilateral Referred to in Algorithm 6a.

Note that the swap test of Algorithm 6a requires only 10 multiplications and 13 additions. This is a considerable improvement over algorithms which compute quantities monotonically related to the angles and do comparisons.

Algorithm 6a may produce an incorrect decision due to floating-point

arithmetic error when $SIN(\alpha_1+\alpha_2)$ is near 0. This can only occur in the neutral case ($\alpha_1+\alpha_2 = \pi$) and when the four vertices are nearly collinear (α_1 and α_2 both near 0 or π). The numerical instability in the neutral case has no ill effects except that the choice of diagonal arcs in a uniform rectangular grid is neither predictable nor consistent, resulting in a displeasing appearance. No attempt is made to remedy this situation in our code.

On the other hand, it is critical that the correct decision be made when the quadrilateral vertices are nearly collinear as the following examples show.

Example 1. Consider the four-node triangulation depicted in Fig. 7a. A perturbation of this triangulation is depicted in Fig. 7b. Node 4 was found to be exterior to $T_3 = \{[1,2,3)\}$; but due to roundoff error, Algorithm 6a might produce the decision to apply a swap to (1,2,3,4), destroying the triangulation. Thus the swap test should be negative (SWPTST = FALSE) when α_1 and α_2 are approximately zero.

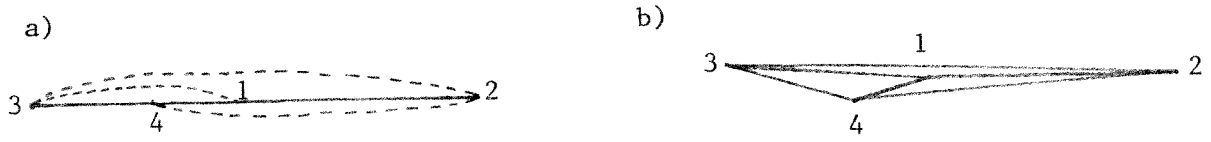


Fig. 7. Triangulation with Nearly Collinear Nodes.

Example 2. In the triangulation depicted in Fig. 8a, Node 4 was found to be interior to $T_3 = \{(1,2,3)\}$ while Node 5 was determined to be exterior to T_4. However, due to roundoff error, Algorithm 6a applied to (2,1,5,4) resulted in the decision not to swap, leaving the triangulation nonoptimal. If a swap had been applied, it would have been followed by a swap applied to (4,1,5,3) resulting in the Thiessen triangulation shown in Fig. 8b. Note that the null or nearly null triangle (5,2,4) will be eliminated if a new

node is added to the right of $5 \rightarrow 2$. Thus the test should be positive when both α_1 and α_2 are approximately equal to π.

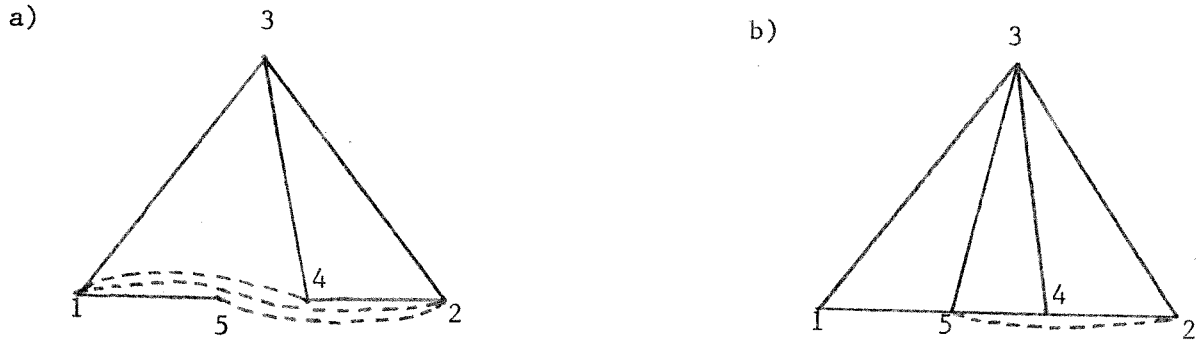


Fig. 8. Triangulation for Example 2.

The following alternative algorithm eliminates this numerical instability and is the method implemented in our software package.

Algorithm 6b. Given the X and Y coordinates of N_1, N_2, N_3, and N_4 where (N_1,N_2,N_3,N_4) is a quadrilateral of T:

Step 1: Set COS1 and COS2 to $(X_1-X_3)(X_2-X_3) + (Y_1-Y_3)(Y_2-Y_3)$ and $(X_2-X_4)(X_1-X_4) + (Y_2-Y_4)(Y_1-Y_4)$, respectively.

Step 2: If COS1 $\geq$ 0 and COS2 $\geq$ 0, go to Step 7.

Step 3: If COS1 < 0 and COS2 < 0, go to Step 6.

Step 4: Set SIN1 and SIN2 to $(X_1-X_3)(Y_2-Y_3) - (X_2-X_3)(Y_1-Y_3)$ and $(X_2-X_4)(Y_1-Y_4) - (X_1-X_4)(Y_2-Y_4)$, respectively. Set SIN12 to SIN1*COS2 + COS1*SIN2.

Step 5: If SIN12 $\geq$ 0, go to Step 7.

Step 6: Set SWPTST to TRUE and stop.

Step 7: Set SWPTST to FALSE and stop.

This algorithm is algebraically equivalent to Algorithm 6a except when $\alpha_1 = \alpha_2 = \pi$. Note that if Step 5 is reached, $\alpha_1 + \alpha_2$ is in the range $(\pi/2, 3\pi/2]$. Thus, assuming the nodes are distinct, numerical instability

occurs only in the neutral case. A further discussion of floating-point errors is found in Renka.[13]

The operation count for Algorithm 6b is:

a) 4 multiplies, 10 adds, and 2 compares, or

b) 4 multiplies, 10 adds, and 4 compares, or

c) 10 multiplies, 13 adds, and 5 compares.

depending upon the values of α_1 and α_2. If α_1 and α_2 are uniformly distributed over the range $[0,\pi]$, the expected operation count is 7 multiplies, 11.5 adds, and 4 compares. In any case there are less arithmetic operations but more compares than in Algorithm 6a. Thus, compares being generally more expensive, we pay a price in efficiency for the numerical stability.

The following algorithm for determining the closest k nodes to a given node is used in our interpolation software to select a set of nodes whose data values are to enter into derivative estimates. It also has application in closest-point problems.

Algorithm 7. Given a Thiessen triangulation T and a node N_0, determine a sequence of nodes $N_1,N_2,\ldots,N_k$ ordered by distance from N_0 for some $k \geq 1$.

Briefly stated, the algorithm is as follows:

Set S_0 to $\{N_0\}$.

For $i = 1,2,\ldots,k$:

Step 1: Mark N_{i-1} (e.g., with a negative pointer to its adjacency set).

Step 2: Set S_i to the union of S_{i-1} with the set of neighbors of N_{i-1}.

Step 3: Set N_i to the unmarked node in S_i which is closest to N_0.

Note that at Step 3 the unmarked nodes in S_i (along with any marked boundary nodes) are the vertices of a simple circuit. See Fig. 9. The following is a detailed description of a method for traversing this circuit (looping through the nodes) based on adjacency information. No extra storage is required for the elements of S_i. The variables are as follows: ND is N_j for some $j < i$, NB is a neighbor of ND, NC is a candidate for N_i, and NST is the starting point of the search.

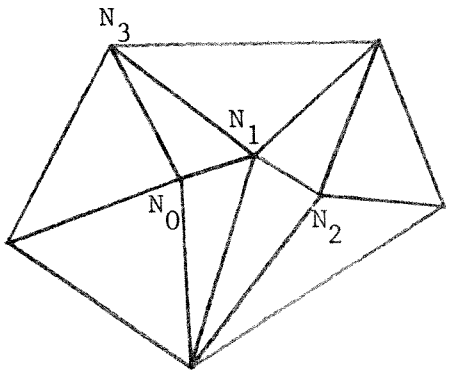


Fig. 9. Portion of T whose Boundary Nodes are the Unmarked Elements of S_3

For i = 1,2,...,k:

Step 1: Mark N_{i-1}, set ND to N_{i-1} and set NB to an unmarked neighbor of ND. (If N_{i-1} is a boundary node, it may not have an unmarked neighbor. In this case try N_{i-1}, N_{i-3}, etc.)

Step 2: Set NST and NC to NB and set DNC to $|N_0 - NC|^2$ (where $|.|$ denotes Euclidean distance).

Step 3: Set NBSAV to NB.

Step 4: If ND is a boundary node and NB is its last neighbor, set NBSAV to ND. Set NB to the next neighbor of ND (following NB in cyclical counterclockwise order).

Step 5: If NB = NST, go to Step 8.

Step 6: If NB is marked, set ND to NB, set NB to NBSAV, and go to Step 4.

Step 7: Set DNB to $|N_0 - NB|^2$. If DNB < DNC, set NC to NB and DNC to DNB. Go to Step 3.

Step 8: Set N_i to NC. [End of loop on i.]

Unmarked nodes $N_0, N_1, \ldots, N_{k-1}$.

Algorithm 7 is proved correct in Renka.[12]

While designed for points in the plane, the above algorithms require only minor modifications to treat alternative geometries. Essentially, by altering only the swap test and definition of LEFT, Renka[14] has extended the triangulation procedure to the surface of the sphere. Similar modifications could be made for the case of nodes on the surface of a cylinder. However, since the algorithms rely on an ordering of the adjacency information, substantial alterations are required to treat data in higher dimensional spaces.

4. DATA STRUCTURES

This section describes various data structures for representing the triangular grid. The choice of data structure involves a trade-off between computational efficiency and storage efficiency. Thus the best method of representing the triangulation depends on the available computing resources and the application. The first two subsections below describe data structures which contain the adjacency information explicitly and are thus well suited for the algorithms described in the previous section. The third subsection discusses Lawson's data structure; and the final subsection contains a table of storage requirements.

4.1 Adjacency Array

The primary goal of this research was a triangulation method requiring less storage than those currently available. This led us to the following data structure which is designed to limit the storage requirement while remaining computationally feasible.

IADJ - Array containing the sequentially ordered set of adjacency lists (indices of the neighbors in the adjacency sets) where the adjacency list of each boundary node is modified by the addition of index 0 following the index of the last neighbor and representing a "pseudo node" infinitely distinct from the boundary. The adjacency list for node k is followed by the list for node k + 1, k = 1,...,N-1.

IEND - Array of length N containing pointers to the ends of each (modified) adjacency list in IADJ.

Thus the indices of the neighbors of Node 1 are stored in IADJ(1),...,IADJ(IEND(1)). For k > 1, the indices of the neighbors of node k are stored in IADJ(IEND(k-1)+1),...,IADJ(IEND(k)), and k has IEND(k) - IEND(k-1) neighbors including (possibly) the pseudo node represented by index 0. Node k is a boundary node if and only if IADJ(IEND(k)) = 0. See Fig. 10

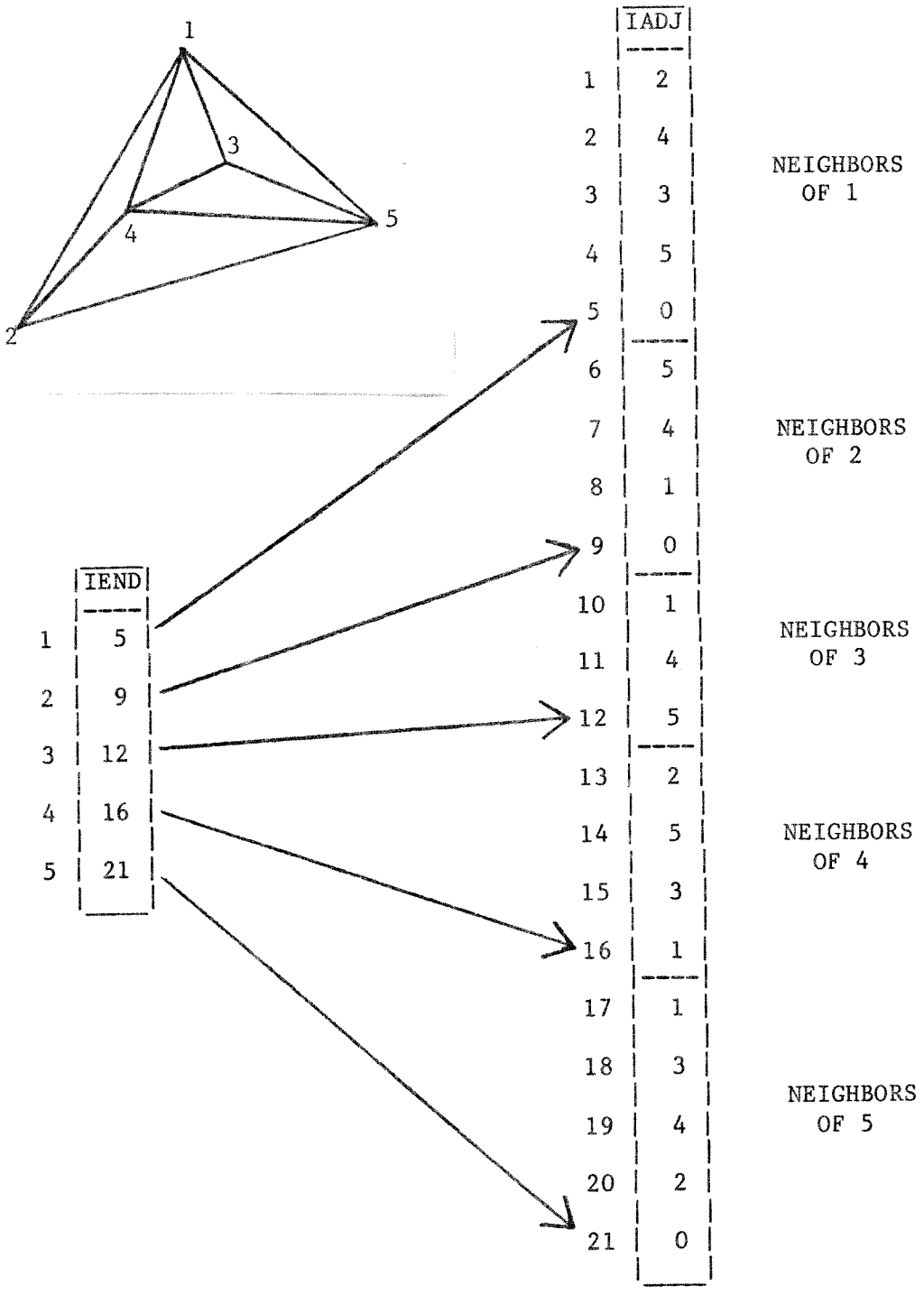


Fig. 10. Sample Triangulation and Adjacency Array.

Let L be the length of IADJ. For each arc of T there is a pair of adjacent nodes, say N_1 and N_2, and exactly two entries in IADJ -- N_1 as a neighbor of N_2 and N_2 as a neighbor of N_1. The adjacency array also contains a zero entry for each node on the boundary. Thus, since there are no other elements in IADJ,

$$L = 2N_a + N_b = 6N - 6 \leq 6N - 9$$

where the second equation follows from Theorem 1 and the inequality is obtained from the lower bound on N_b. The upper bound on L determines the amount of storage which must be reserved since the value of N_b is generally not known before the triangulation is constructed.

Clearly there is some redundancy in the adjacency array in that for each arc $N_1 - N_2$, both N_1 as a neighbor of N_2 and N_2 as a neighbor of N_1 are represented explicitly. Thus we have not attempted to minimize the storage requirement. However, the total requirement of less than 7N locations represents a substantial savings over other available triangulation methods as shown in Table 1, Section 4.4.

This storage efficiency was gained at a cost in computational efficiency caused by the necessity of shifting portions of the adjacency array up or down for deletions and insertions of neighbors as the triangulation is constructed. Timing comparisons are presented in Section 5. We feel this trade-off of computational efficiency for storage efficiency is generally advantageous since it allows larger problems to be solved on small machines (even micro-computers), and the time required to construct the triangulation is usually insignificant relative to the time spent on interpolation which is the primary application. Thus we have chosen to employ the adjacency array in our software package.

An obvious modification to the above data structure is the replacement of IEND with an array ISTART of pointers to the beginnings of each adjacency list in IADJ. However, in order that the index of the last neighbor of each node be easily accessible, ISTART must have length N + 1 with ISTART(N+1) pointing to the first empty location in IADJ. Then the index of the last neighbor of Node k (or 0 representing the boundary) is stored in IADJ(ISTART(k+1)-1) for k = 1,...,N.

4.2 Linked List

The following alternative data structure is not used in our software package but was implemented for comparison with the adjacency array. The linked list eliminates the necessity of shifting portions of an array for insertions and deletions. Rather than storing adjacency lists and their elements in contiguous locations, the index of each neighbor has an arbitrary location in the list with a list pointer to the index of the neighbor which follows it in cyclical counterclockwise order. The method for representing the boundary has also been modified. The linked list consists of three arrays and a pointer:

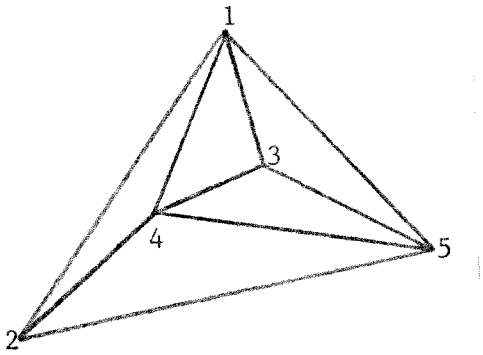
LIST - Array containing the indices of the neighbors of node k for k = 1,...,N. LIST contains the negative index of the last neighbor of a boundary node.

LPTR - Array of LIST pointers in a one-to-one correspondence with the elements of LIST. The LIST pointer associated with node j as a neighbor of k points to the index of the neighbor of k which follows j in cyclical counterclockwise order.

LEND - Array of length N containing a LIST pointer to the index of the last neighbor of each node.

LNEW - Pointer to the first empty location in LIST and LPTR.

Note that k is a boundary node if and only if (LIST(LEND(k)) < 0. See Fig. 11.



	LIST	LPTR
1	2	3
2	3	11
3	4	2
4	-1	14
5	1	7
6	1	10
7	4	12
8	4	4
9	3	6
10	2	13
11	-5	1
12	5	5
13	5	9
14	5	8
15	1	16
16	3	17
17	4	18
18	-2	15

	LEND
1	11
2	4
3	7
4	10
5	18

LNEW = 19

Fig. 11. Sample Triangulation and Linked List.

We have chosen to store pointers to last neighbors (LEND) rather than first neighbors because it is convenient to have easy access to both first and last neighbors of a boundary node. To determine the index of the last neighbor of node k starting from the LIST pointer to its first neighbor, it is necessary to follow pointers through the entire adjacency list for k; whereas the index of the first neighbor is readily obtained from a pointer to the last neighbor. The elimination of index 0 representing the boundary

allows more efficient access to first neighbors at the cost of occasionally having to change signs of indices.

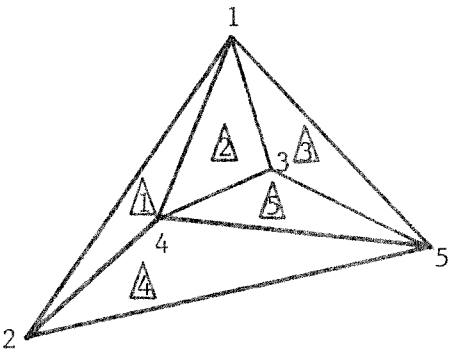
By Theorem 1, the length of LIST and LPTR is $2N_1 = 6N - 2N_b - 6 \leq 6n - 12$, implying a total storage requirement of less than 13N locations. Thus with the adjacency array replaced by the linked list, our triangulation method still requires significantly less storage than most other available methods while being comparable to other methods with regard to computational efficiency. See Table 1, Section 4.4 and Table 2, Section 5.

Unlike the adjacency array, the linked list is well suited for packing: storing more than one integer per computer word with storage and retrieval achieved by shifts or arithmetic operations. In an implementation on a machine with sufficiently large word length, an element of LIST and its associated LIST pointer may be conveniently stored in a single word, resulting in a savings of 6N storage locations.

4.3 Lawson's Data Structure

We present a description of the data structure used by Lawson[5] for comparison with those previously discussed. It consists of a list containing six integers for each triangle. The first three integers are the indices of adjacent triangles in counterclockwise order with 0 representing the region exterior to the convex hull H. The last three integers are the triangle's vertex indices in counterclockwise order, the first vertex being the node shared by the first and third adjacent triangles. See Fig. 12.

This data structure has an advantage over the adjacency array in that no shifting or garbage collection is necessary; i.e., triangles need never be deleted -- only replaced. Thus updating the data structure for a swap or the addition of a new node can be implemented very efficiently. On the other hand, the ordered sequence of boundary nodes is not readily obtained



TRIANGLE INDEX	INDICES OF ADJACENT TRIANGLES IN COUNTER-CLOCKWISE ORDER - A ZERO INDICATES THE REGION EXTERIOR TO H			VERTEX INDICES IN COUNTER-CLOCKWISE ORDER - THE FIRST VERTEX IS SHARED BY THE FIRST AND THIRD ADJACENT TRIANGLES		
1	2	0	4	4	1	2
2	5	3	1	4	3	1
3	5	0	2	3	5	1
4	1	0	5	4	2	5
5	4	3	2	4	5	3

Fig. 12. Sample Triangulation and Lawson's Data Structure.

from the triangle list; and, in order to determine the set of nodes which are visible from an exterior point, Lawson uses an additional array of length $4N_b$.

Lawson's algorithm also requires two arrays of length N used to sort the nodes by their distances from an initially determined boundary node so that a new node to be added to the triangulation is always exterior to the convex hull of the previously added nodes.

From Theorem 1, $N_t \leq 2N - 5$ and hence the total storage requirement for Lawson's method is $6N_t + 2N + 4N_b \leq 14N + 4N_b - 30$.

An advantage of the triangle list is the fact that it lends itself to convenient packing of either two or three integers per computer word. Lawson's method has been implemented with three integers packed into a 36-bit word. However, the number of nodes is limited to 2050 in this case. See Table 1.

4.4 Storage Requirements

The following table compares the storage requirements of our triangulation method (using both the adjacency array and the linked list) with the methods of Lawson,[5] Akima,[9] Shamos,[10] and Green and Sibson.[11] The specified requirements are in addition to the 2N locations containing nodal coordinates. Note that the number of boundary nodes N_b cannot generally be predicted, and thus the upper bound of N must be used in reserving storage.

Table 1. Triangulation Storage Requirements

Adjacency Array	Linked List	Lawson	Akima	Shamos	Green and Gibson
7N	13N	$14N + 4Nb \leq 18N$	20N	30N	$\geq$ 11N

The storage required for the linked list and Lawson's method may be reduced to 7N and 12N, respectively, by packing two integers per computer word. Lawson's method also allows three integers to be packed in a word reducing the requirement to 10N. Note, however, that packing is machine-dependent. Green and Sibson use a heap along with garbage collection, thus allowing extra storage to be employed for increased efficiency. The specified storage requirement is the minimum amount which allows reasonable time efficiency.

5. TIMING COMPARISONS

We have determined timing requirements for various triangulation methods, domains, and values of N. Lawson's triangulation software was the only code available to us other than our own, but it is the most efficient method that we are aware of. The times specified in Table 2 are central-processor seconds obtained on the CDC Cyber 170-75 at the University of Texas at Austin using the MNF (Minnesota FORTRAN) compiler. The timings were averaged over a sufficient number of runs to eliminate random errors in the timer. In the following table p denotes the log (base 2) of R where R is the ratio of the time associated with N = 2000 to that associated with N = 1000; i.e., the times associated with the two values of N were fit with the model CN^p.

We conclude that Lawson's method is faster than any of the others, but only slightly faster than the linked list with reordering of the nodes. Also, the growth rates for Lawson's method are smaller than those of the other methods. For both the adjacency array and the linked list, reordering of the nodes is advantageous. Critical values of N at which reordering becomes advantageous have been found to be less than 100 in all cases, as the growth rates would indicate.

Table 2. Timing Requirements for Triangulation (and Reordering) of N Randomly Generated Points

Disc of Unit Radius

Method	N		p
	1000	2000	
Lawson	1.86	4.05	1.12
Adjacency Array With Reordering	2.91	7.09	1.29
Adjacency Array (No Reordering)	15.24	57.24	1.91
Linked List With Reordering	1.89	4.26	1.17
Linked List (No Reordering)	2.78	6.77	1.29

Unit Square

Method	N		p
	1000	2000	
Lawson	1.83	3.97	1.12
Adjacency Array With Reordering	3.08	7.47	1.28
Adjacency Array (No Reordering)	15.56	57.33	1.88
Linked List With Reordering	1.95	4.37	1.16
Linked List (No Reordering)	1.81	6.90	1.29

REFERENCES

1. R. J. Renka, A Triangle-Based C^1 Interpolation Method, ORLN/CSD-103, Oak Ridge National Laboratory, 1982.

2. M. I. Shamos and D. Hoey, "Closest-Point Problems," Proceedings 16th Annual Symposium on Foundations of Computer Science, pp. 151-62, 1975.

3. R. B. Simpson, "A Two-Dimensional Mesh Verification Algorithm," SIAM J. Sci. Stat. Comput, Vol. 2, No. 4, Dec. 1981.

4. B. G. Ryder, "The PFORT Verifier," Software Practice and Experience, Vol. 4, pp. 359-77, 1974.

5. C. L. Lawson, "Software for C^1 Surface Interpolation," Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 161-94.

6. C. L. Lawson, Generation of a Triangular Grid with Applications to Contour Plotting, Technical Memorandum 299, California Institute of Technology Jet Propulsion Laboratory, 1972.

7. R. Sibson, "Locally Equiangular Triangulations," Computer Journal, Vol. 21, No. 3, pp. 243-45, 1978.

8. D. Rhynsburger, "Analytic Delineation of Thiessen Polygons," Geographical Analysis, Vol. 5, No. 2, pp. 133-44, 1973.

9. H. Akima, "A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points," ACM TOMS, Vol. 4, No. 2, pp. 148-64, 1978.

10. M. I. Shamos, Computational Geometry, PhD dissertation, Yale University, 1978.

11. P. J. Green and R. Sibson, "Computing Dirichlet Tessellations in the Plane," Computer Journal, Vol. 21, No. 2, pp. 168-73, 1978.

12. R. J. Renka, A Storage-Efficient Method for Construction of a Thiessen Triangulation, ORNL/CSD-101, Oak Ridge National Laboratory, 1982.

13. R. J. Renka, Triangulation and Bivariate Interpolation for Irregularly Distributed Data Points, PhD dissertation, University of Texas at Austin, 1981.

14. R. J. Renka, Interpolation of Data on the Surface of a Sphere, ORNL/CSD-108, Oak Ridge National Laboratory, 1982.

ADDENDUM TO TR-213

Page	Line	Changes
2	4	Renka[8] => Renka[13]
2	6	requiring less than => requiring less storage than
3	-2	(N_i,N_j,N_j) with N_k STRICTLY LEFT $N_i \rightarrow N_j$ for $i,j,k \in \{1,2,3\}$ => (N_i,N_j,N_k) with N_k STRICTLY LEFT $N_i \rightarrow N_j$ for $(i,j,k) \in \{(1,2,3), (2,3,1), (3,1,2)\}$
4	4	b) The interiors => b) the interiors
5	-3	form a quadrilateral of T => form a quadrilateral of T
7	13	Either $N_1 - N_2$ or N_3 may be => Either diagonal may be
7	-5	referred to in the literatures as => referred to in the literature as
9	5	Algorithms 3, 4, and 7 are => Algorithms 3 and 5 are
9	-4	One of the features of this method is that => One of the advantages of this method over alternatives is that
11	2 to 3	let H be the convex hull, let B be its boundary and let P => let H be the convex hull of S with boundary B, and let P
12	5	and go to step 5. => and go to step 7; otherwise set N_2 to N_1 and go to step 5.
14	-3	then swap N_1 => then swap $N_1 - N_2$
15	-5	N_2 N_3 N_1 and => (N_2 N_3 N_1) and
15	-5	N_1 N_4 N_2 (ref. Fig. 6). => (N_1 N_4 N_2) (ref. Fig. 6).
15	-4	α_1 => α_2
16	7	<<all four '1's should read '1/2's>>
16	9	<<all four '1's should read '1/2's>>
16	Fig. 6	<<bottom right 'N_3' should read 'N_4'>>
17	13	{[1,2,3)}; => {(1,2,3)};
19	6	compares. => compares,

Page	Line	Changes
20	3 to end	<<delete from "Note that at ... counterclockwise order).">>
21	1 to 7	<<delete from "Step 5: ... correct in Renka.[12]">>
22	-11	distinct => distant
22	last	Fig. 10. => Fig. 9.
23	last	Fig. 10. => Fig. 9.
24	6	$6N - 6$ => $6N - N_b$
25	last	Fig. 11. => Fig. 10.
26	-8	Fig. 11. => Fig. 10.
27	3	LPTR is $2N_l = 6N - 2N_b - 6 \leq 6n$ => LPTR is $2N_a = 6N - 2N_b - 6 \leq 6N$
27	-6	See Fig. 12. => See Fig. 11.
28	-10	Fig. 12. => Fig. 11.
32	2	ORLN => ORNL
32	-10	1978. => 1975.